Ethereum, Hyperledger fabric and R3 Corda

The major intention of this paper is to analyse the three Distributed Ledger Technologies (DLTs) Ethereum, Hyperledger Fabric and R3 Corda and provide an overview of the individual DLTs to users new to DLTs. The paper dives into the various aspects of the three DLTs, namely: the programming language used to create applications and contracts for the DLTs, their architecture, governance, smart contracts, consensus algorithm, scalability and privacy issues, currency and use cases, in order to showcase the major technical differences among the three DLTs.

Ethereum was developed as a permissionless public Blockchain where anyone can write smart contracts and decentralised applications using the built in programming language called Solidity. The development of Ethereum was made independent of any specific field of application.

Hyperledger fabric is a permissioned, shared ledger. The admission of participants are managed due to the permissioned nature of Hyperledger. The basic architecture of Hyperledger is based on multiple ledgers whose operations are independent of one another but there exists an addressing system that allows a transaction of one ledger to discover and utilise the transactions and smart contracts on another ledger. Hyperledger provides an extendable and modular architecture that can be employed for various fields and hence is made independent of any specific filed of application.

R3 Corda is a Global Logical ledger in which all Economic actors interact allowing parties to record and manage agreements amongst themselves in a secure, consistent, reliable, private and authoritative manner. The word global in the global logical ledger is put in the sense that everyone sees the same data that pertains to them and the logical part is to signify that the physical implementation may be composed differently. Unlike Hyperledger Fabric and Ethereum, R3 Corda was developed solely envisioning the Financial industry.

Ethereum

Language

The built in programming language for Ethereum is Solidity. It is a high level language for implementing smart contracts to be executed on the Ethereum Blockchain. The language is influenced by C++, Python and Java Scripts and is implemented on the Ethereum Virtual Machine (EVM). The major advantage of solidity is its ease of understanding and implementation.

A 'Go' implementation for writing Ethereum smart contracts is available but the contract written in golang cannot be directly implement them on the EVM. The programmer has to write his own compiler to convert the smart contracts in golang to EVM bytecode.

Architecture

Ethereum, being an open and permissionless Blockchain, any developer can build an application using the inbuilt programming language to interact with the Blockchain. These applications care called Decentralized applications or Dapps. The users use these Dapps to

interact with the blockchains. Miners create blocks in which user transactions are hashed and ordered. The node operators validate miner blocks and process users' transactions on those blocks.

Governance

There is no centralised governance in Ethereum but is distributed among the various parties based on incentive. The Full Node operators decide which software their nodes will run. Miner operators decide whether to mine in a mining pool or to mine solo. Geth and Parity Codebases (command line interface used to run nodes) make their own decisions to maintain cross-client compatibility and have their representative developers attend the 'All devs' call where they have a direct conversation with the representatives of the Ethereum foundation.

Smart Contract

A smart contract is a computer protocol which is intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract between two or more network entities (nodes). The term 'contract' in Ethereum does not signify something that should be 'fulfilled' or 'complied with'; rather an 'autonomous agent' living inside the Ethereum execution environment, always executing a specific piece of code when triggered by a message or transaction. The code consists of a series of bytes, with each byte signifying an operation. The Smart Contracts in Ethereum run in the Ethereum Virtual Machine (EVM). An Ethereum transaction contract code can trigger data reads and writes, do computations like using cryptographic primitives, make calls (send messages) to other contracts, etc.

Consensus

In a Distributed Ledger Technology, consensus among the nodes must be reached to validate a transaction. The current instance of Ethereum uses the Proof of Work (PoW) consensus algorithm. PoW uses a 'hash function' to create conditions. A participant solves the cryptographic puzzles and creates new blocks (mining). These blocks are then independently verified by other system participants.

Ethereum's upcoming Casper implementation will introduce the Proof of Stake (PoS) consensus algorithm. The PoS algorithm is similar to the PoW system but the participants in the consensus building process is restricted to parties who have been identified to be having a legitimate stake in the Blockchain. The PoS removes the hash function calculation with a simple digital signature that proves ownership of stake. This would reduce distribution of information among nodes who have no legitimate stake in the Blockchain.

Scalability

Ethereum is a permissionless, open-source Blockchain that operates on the PoW consensus algorithm. The participants involved in the consensus algorithm are not restricted and that leads to scalability issues for the network.

The change to PoS algorithm would mean limited participants in the consensus algorithm, increasing the scalability of Ethereum.

Privacy

The permissionless nature of Ethereum means that, irrespective of legitimate stake in the Blockchain, a node involved in the consensus algorithm gets a copy of the transaction. This leads to privacy issues in the Ethereum Blockchain.

Similar to the scalability issue, the change to PoS algorithm will also solve the privacy issue by restricting participation in the consensus algorithm.

Currency

As discussed above, the smart contracts in Ethereum are executed in the EVM. Every single operation that are executed inside the EVM is simultaneously executed by every node in the network. The existence of *gas,* the internal pricing system for running a transaction or contract in Ethereum, is to limit the usage of resources by each contract. Each operation triggered by a contract based on cost measured in gas and each gas unit consumed by a transaction is paid for in Ether based on the current gas/Ether prices (this changes continuously).

Use Case in Finance

Daimler has partnered with Landesbank Baden-Württemberg (LBBW) to sell a relatively short-term one-year bond to the savings banks Esslingen-Nürtingen, Ludwigsburg and Ostalb. The bond was issued on a private version of the Ethereum Blockchain.

Hyperledger Fabric

Language

Contracts in Hyperledger are known as Chaincode. Hyperledger supports writing chaincode in Golang or Java language. The chaincode support for Java is in Beta. The chaincodes are finally executed inside a docker container.

Architecture

Hyperledger facric is a blockchain framework implementation by the Linux Foundation.

The permissioned nature of Hyperledger is created with the help of the issuing authority on the network. Validating and non-validating nodes are run by white listed organisations and transactors. These organisations and transactors are granted identity by the issuing authority. Depending on the purpose, the issuing authority assigns the appropriate level of access required to obtain an identity and transact on the network converting. Due to the varying level of access of each participants, Hyperledger is a permissioned ledger.

The permissioned nature of Hyperledger arose from the need for privacy of operation for the participants. But the need for privacy does not exclude the need for identification and audit ability from regulators. Hence, the encryption of identity is done in such a way that it remains concealed from other unwanted participants but can be accessed by the regulators.

Hyperledger starts with cryptographic certificate encapsulating a user's confidential data, which is registered in a registration authority. From each identity, the protocol can generate security keys for members to transact on a network, which conceal the identities of the transacting parties, providing privacy support to the network.

Within Hyperledger, content confidentiality is achieved by encrypting the transactions such that only the stakeholders can decrypt and execute them. In addition, a piece of business logic (realised by a smart contract) can also be cryptographically secured (if confidentiality is required by its stakeholders) so that it only gets loaded and decrypted at runtime.

Governance

Hyperledger is a private validator network protocol, meaning that all entities are required to register with membership services to obtain an identity with access and transaction authority on the network. During the network setup, validators can determine the level of permission that is required to transact. The Network set-up also defines the network as permissive, allowing ease of access and support for rapid and high adoption, or restrictive for a more controlled environment.

Hyperledger fabric is a permissioned network, means that the control of governance lies with the Linux Foundation and the organisations within the blockchain.

Smart Contract paradigm

A chaincode is a decentralized transactional program, running on the validating nodes. Chaincode transactions are time bounded and configured during chaincode deployment, which is similar to a database call or a Web service invocation. If a transaction times out, it is considered as an error and will not cause state changes on the Ledger. One chaincode function can call another chaincode function if the callee has the same restrictive confidentiality scope; that is, a confidential chaincode can call another confidential chaincode if they share the same group of validators.

As transactions are run in a new block, a delta from the world state in the last block on the blockchain is maintained. If consensus is reached for the current block, the changes are committed to the database, and the world state block number is incremented by 1. If peers do not reach consensus, the delta is discarded and the database is not modified.

Consensus

Consensus algorithms under Hyperledger are pluggable, allowing users to select the algorithm of their choice during deployment. The Hyperledger protocol will provide an implementation of Byzantine Fault Tolerance (BFT) in its initial release, using the PBFT protocol [CL02].

Scalability

The Hyperledger network structure is permissioned, controlling the number of participants in each Blockchain. The controlled number of participants increases the scalability of Hyperledger.

Privacy

The issuing authority of Hyperledger encrypts the identity of each user and the transactions made by the users. The encrypted data can only be decrypted at run-time by the required participants hence solving the privacy issues.

Currency

Hyperledger Fabric does not have a native currency. A common attribute of a currency is that an amount will be transacted every time a transaction is processed on its chain. Based on the requirement of individual organisations, currencies on Hyperledger can be created using chaincodes.

Use Case in Finance

Hyperledger Fabric's trade finance platform aimed at international payments utilising Blockchain technology and are supported by Deutsche Bank, HSBC, KBC, Natixis, Robobank, Societe Generale and Unicredit.

R3 Corda

Language

The code in Corda is written using Kotlin, a programming language from JetBeans that targets the JVM and Javascript. The major reason for chosing Kotlin is the high level of integration. Due to the JVM, any related programming paradigm can be used.

Architecture

The major goal of corda is to create a 'global logical ledger' in which all economic actors will interact allowing parties to record and manage agreements amongst themselves in a secure, consistent, reliable, private and authoritative manner. The word global in the global logical ledger is put in the sense that everyone sees the same data that pertains to them and the logical part is to signify that the physical implementation may be composed differently. As such, a possible end-state is one in which we have moved from authoritative systems-of-record

maintained within firms to global authoritative systems-of-record shared between firms making it not centralised by distributed.

The architecture and strategic choices of corda for recording and proceeding financial agreements includes three major visions: firstly, records managed by this system will be accessible only to those actors with a legitimate interest in the assets and agreements they manage. Secondly, the behaviour of agreements managed by the system will be described in computer code that explicitly refers to and gains its legitimacy from over-arching legal prose. And finally, to gain wide adoption across the financial community, portions of the system must and will be open: open source, open development process, open, technological industry standards.

The key activities or features of Corda, as stated in the Corda Whitepaper, includes:

· Recording and managing the evolution of financial agreements and other shared data between two or more identifiable parties in a way that is grounded in existing legal constructs and compatible with existing and emerging regulation.

· Choreographing work flow between firms without a central controller.

· Supporting consensus between firms at the level of individual deals, not a global system.

· Supporting the inclusion of regulatory and supervisory observer nodes.

· Validating transactions solely between parties to the transaction.

· Supporting a variety of consensus mechanisms.

· Recording explicit links between human-language legal prose documents and smart contract code.

· Using industry-standard tools.

· Restricting access to the data within an agreement to only those explicitly entitled or logically privileged to it.

Corda began as a single global ledger. In case when the transaction involves a small subgroup of parties, corda strives to keep the relevant data purely within the subgroup.

The foundation object is a state object that records the existence, content and current state of an agreement between two or more parties. It is intended to be shared only with those who have a legitimate reason to see it. To ensure consistency in a global, shared system where not all data is visible to all participants, Corda relies heavily on secured cryptographic hashes to identify parties and data. The ledger is defined as a set of immutable state objects.

The modularity and interoperability of Corda enables organisations to integrate already existing setup, such as databases, into the Corda network.

Governance

Corda is a permissioned blockchain which gives the control of Governance to R3 and the organisations participating in the transaction.

Smart Contract

Corda supports smart contracts. Smart contracts in Corda are agreement whose execution is both automatable by computer code working with human input and control, and whose rights and obligations, as expressed in a legal prose, are legally enforceable. The smart contract links business logic and business data to an associated legal prose in order to ensure that the financial agreements on the platform are rooted firmly in law and can be enforced in the event of ambiguity, uncertainty or dispute.

Corda enforces business logic through smart contract code, which is constructed as a pure function that either accepts or rejects a transaction, and which can be composed from simpler, reusable functions.

Contracts define a part of the business logic on the ledger, and they are mobile: Nodes will download and run contracts inside a sandbox without any review in some deployments, although Corda envisages the use of signed code for Corda deployments in the regulated sphere.

The virtual machine selected for contract execution and validation is the Java Virtual Machine. However, virtual machine has been augmented with a custom sandbox that is radically more restrictive than the ordinary JVM sandbox, and it enforces not only security requirements but also deterministic execution.

Consensus

In Corda, there are two aspects of consensus:

· Transaction validity: parties can reach certainty that a proposed update transaction defining output states is valid by checking that the associated contract code runs successfully and has all the required signatures; and that any transactions to which this transaction refers are also valid.

· Transaction uniqueness: parties can reach certainty that the transaction in question is the unique consumer of all its input states. That is: there exists no other transaction, consensus have been reached (validity and uniqueness), that consumes any of the same states.

Parties can agree on transaction validity by independently running the same contract code and validation logic.

A notary is a network service providing uniqueness consensus for a given transaction. Notary provides the point of finality in the system. Parties cannot be sure that an equally valid, but conflicting transaction is regared as a valid attempt to spend the given input state until the notary signature is obtained. Each state has an appointed notary, and the notary will only notarise the transaction if it is the appointed notary of all the transaction's input states.

Corda has 'pluggable' uniqueness services. This is to improve privacy, scalability, legal-system compatibility and algorithmic agility. A single service may be composed of many mutually non-trusting nodes coordinating via a byzantine fault tolerant algorithm, or could be very simple, like a single machine. In some cases, like when evolving a state requires the signatures of all relevant parties, there may be no need for a uniqueness service at all.

These uniqueness services are required only to attest whether the states consumed by a given transaction have previously been consumed; they are not required to attest as to the validity of the transaction itself, which is a matter for the parties to the transaction. This means that the uniqueness services are not required to (and, in the general case, will not) see the full contents of any transactions, significantly improving privacy and scalability of the system compared with alternative distributed ledger and blockchain designs. This design decision represents an important choice as to the acceptable trade-offs in shared ledger architectures.

Scalability and Privacy

The pluggable uniqueness service in Corda and the use of shared cryptographic hashes to ensure restrictive viewing of transactions tackle the scalability and privacy issues.

Currency

Corda does not have an internal currency but will support currencies in the future.

Use Case in Finance

A group of 11 banks have developed a trade finance application on Corda. The application aims at smoothing the process for letters of credit. The banks involved include Bangkok Bank, BBVA, BNP Paribas, HSBC, ING, Intesa Sanpaolo, Mizuho, RBS, Scotiabank, SEB and U.S. Bank.

'Code is Law'

All executions is Ethereum is final. All the transactions are immutable. Even in case of conflicts and disagreements between two parties, the transaction cannot be reverted.

Corda has a roll back function. If there is an accidental error, the incorrect data can be presented to a non validating notary that can roll back the transaction.

In case of Hyperledger Fabric, no clear information is available regarding the approach taken to resolve conflicts.

Summary

Characteristics	Ethereum	Hyperledger Fabric	R3 Corda
Programming Language	Solidity	Go, Java	Kotlin
Governance	Distributed among all participants	Linux foundation and organisation in the Chain	R3 and organisations involved.
Smart Contract	Not legally bounded	Not legally bounded	Legally bounded
Consensus Algorithm	PoW. Casper implementation PoS.	PBFT	Notary nodes can run several consensus algorithm
Scalability	Existing scalability issue	Not prevalent	Not prevalent
Privacy	Existing privacy issue	Not prevalent	Not prevalent
Currency	Ether	None Can be made using chaincode	None

Ethereum and Hyperledger fabric are flexible whereas Corda was designed specifically keeping the Financial Industry in focus. Ethereum's programming language Solidity was designed to be easily understandable and simple for implementation. But Ethereum's permissionless mode of operation and simplicity of understanding of Solidity leads to scalability and privacy issues. Fabric solves performance scalability and privacy issues by permissioned mode of operation and specifically by using a BFT algorithm and fine-grained access control. Corda, designed specifically for the finance industry tackles the issues of privacy and scalability but reaching a consensus on the participants involved in the transaction level and not inside the distributed ledger level making it better suited for the Financial industry.